ML Project

1 First approach to the dataset

The project task consists of a binary classification problem. The goal is to perform fingerprint spoofing detection, i.e., to identify genuine vs. counterfeit fingerprint images. The dataset consists of labeled samples corresponding to the genuine (True, label 1) class and the fake (False, label 0) class. The samples are computed by a feature extractor that summarizes high-level characteristics of a fingerprint image. The data is 6-dimensional.

The training files for the project are stored in file `Project/trainData.txt`. The format of the file is the same as for the Iris dataset, i.e., a CSV file where each row represents a sample. The first 6 values of each row are the features, whereas the last value of each row represents the class (1 or 0). The samples are not ordered.

Load the dataset and plot the histogram and pair-wise scatter plots of the different features. Analyze the plots.

1. Analyze the first two features. What do you observe? Do the classes overlap? If so, where? Do the classes show similar mean for the first two features? Are the variances similar for the two classes? How many modes are evident from the histograms (i.e., how many "peaks" can be observed)?

2. Analyze the third and fourth features. What do you observe? Do the classes overlap? If so, where? Do the classes show similar mean for these two features? Are the variances similar for the two classes? How many modes are evident from the histograms?

3. Analyze the last two features. What do you observe? Do the classes overlap? If so, where? How many modes are evident from the histograms? How many clusters can you notice from the scatter plots for each class?

2 PCA and LDA application to the dataset

Apply PCA and LDA to the project data. Start by analyzing the effects of PCA on the features. Plot the histogram of the projected features for the 6 PCA directions, starting from the principal (largest variance). What do you

observe? What are the effects on the class distributions? Can you spot the different clusters inside each class?

Apply LDA (1-dimensional, since we have just two classes), and compute the histogram of the projected LDA samples. What do you observe? Do the classes overlap? Compared to the histograms of the 6 features you computed in Laboratory 2, is LDA finding a good direction with little class overlap?

Try applying LDA as a classifier. Divide the dataset into model training and validation sets (you can reuse the previous function to split the dataset). Apply LDA, select the orientation that results in the projected mean of class True (label 1) being larger than the projected mean of class False (label 0), and select the threshold as in the previous sections, i.e., as the average of the projected class means. Compute the predictions on the validation data, and the corresponding error rate.

Now try changing the value of the threshold. What do you observe? Can you find values that improve the classification accuracy?

Finally, try pre-processing the features with PCA. Apply PCA (estimated on the model training data only), and then classify the validation data with LDA. Analyze the performance as a function of the number of PCA dimensions m. What do you observe? Can you find values of m that improve the accuracy on the validation set? Is PCA beneficial for the task when combined with the LDA classifier?

3 Logdensity

Try fitting uni-variate Gaussian models to the different features of the project dataset. For each component of the feature vectors, compute the ML estimate for the parameters of a 1D Gaussian distribution. Plot the distribution density (remember that you have to exponentiate the log-density) on top of the normalized histogram (set `density=True` when creating the histogram, see Laboratory 2). What do you observe? Are there features for which the Gaussian densities provide a good fit? Are there features for which the Gaussian model seems significantly less accurate?

Note: for this part of the project, since we are still performing some preliminary, qualitative analysis, you can compute the ML estimates and the plots either on the whole training set. In the following labs we will employ the densities for classification, and we will need to perform model selection, therefore we will re-compute ML estimates on the model training portion of the dataset only (see Laboratory 3).

4 Working on Training and Validation sets: PCA and LDA application

Apply the MVG model to the project data. Split the dataset into model training and validation subsets (important: use the same splits for all models, including

those presented in other laboratories), train the model parameters on the model training portion of the dataset and compute LLRs

$$s(\mathbf{x}_t) = \text{llr}(\mathbf{x}_t) = \frac{f_{\mathbf{X}|C}(\mathbf{x}_t|1)}{f_{\mathbf{X}|C}(\mathbf{x}_t|0)}$$

(i.e., with class True, label 1 on top of the ratio) for the validation subset. Obtain predictions from LLRs assuming uniform class priors $P(C = 1) = P(C = 0) = 1/2$. Compute the corresponding error rate (suggestion: in the next laboratories we will modify the way we compute predictions from LLRs, we therefore recommend that you keep separated the functions that compute LLRs, those that compute predictions from LLRs, and those that compute error rate from predictions).

Apply now the tied Gaussian model, and compare the results with MVG and LDA. Which model seems to perform better? Finally, test the Naive Bayes Gaussian model. How does it compare with the previous two?

Let's now analyze the results in light of the characteristics of the features that we observed in previous laboratories. Start by printing the covariance matrix of each class (you can extract this from the MVG model parameters). The covariance matrices contain, on the diagonal, the variances for the different features, whereas the elements outside of the diagonal are the feature co-variances. For each class, compare the covariance of different feature pairs with the respective variances. What do you observe? Are co-variance values large or small compared to variances? To better visualize the strength of co-variances with respect to variances we can compute, for a pair of features i, j, the Pearson correlation coefficient, defined as:

$$\text{Corr}(i,j) = \frac{\text{Cov}(i,j)}{\sqrt{\text{Var}(i)} \cdot \sqrt{\text{Var}(j)}}$$

Or, in matrix form:

$$\text{Corr} = \frac{C}{\left(\text{vcol}(C.\text{diagonal}()^{0.5}) \cdot \text{vrow}(C.\text{diagonal}()^{0.5})\right)}$$

where C is a covariance matrix. The correlation matrix has diagonal elements equal to 1, whereas out-of-diagonal elements correspond to the correlation coefficients for all feature pairs, with

$$-1 \leq \text{Corr}(i,j) \leq 1$$

When $\text{Corr}(i,j) = 0$, the features i, j are uncorrelated, whereas values close to ± 1 denote strong correlation.

Compute the correlation matrices for the two classes. What can you conclude on the features? Are the features strongly or weakly correlated? How is this related to the Naive Bayes results?

The Gaussian model assumes that features can be jointly modeled by Gaussian distributions. The goodness of the model is therefore strongly affected by

the accuracy of this assumption. Although visualizing 6-dimensional distributions is unfeasible, we can analyze how well the assumption holds for single (or pairs) of features. In Laboratory 4 we separately fitted a Gaussian density over each feature for each class. This corresponds to the Naive Bayes model. What can you conclude on the goodness of the Gaussian assumption? Is it accurate for all the 6 features? Are there features for which the assumptions do not look good?

To analyze if indeed the last set of features negatively affects our classifier because of poor modeling assumptions, we can try repeating the classification using only feature 1 to 4 (i.e., discarding the last 2 features). Repeat the analysis for the three models. What do you obtain? What can we conclude on discarding the last two features? Despite the inaccuracy of the assumption for these two features, are the Gaussian models still able to extract some useful information to improve classification accuracy?

In Laboratory 2 and 4 we analyzed the distribution of features 1-2 and of features 3-4, finding that for features 1 and 2 means are similar but variances are not, whereas for features 3 and 4 the two classes mainly differ for the feature mean, but show similar variance. Furthermore, the features also show limited correlation for both classes. We can analyze how these characteristics of the features distribution affect the performance of the different approaches. Repeat the classification using only features 1-2 (jointly), and then do the same using only features 3-4 (jointly), and compare the results of the MVG and tied MVG models. In the first case, which model is better? And in the second case? How is this related to the characteristics of the two classifiers? Is the tied model effective at all for the first two features? Why? And the MVG? And for the second pair of features?

Finally, we can analyze the effects of PCA as pre-processing. Use PCA to reduce the dimensionality of the feature space, and apply the three classification approaches. What do you observe? Is PCA effective for this dataset with the Gaussian models? Overall, what is the model that provided the best accuracy on the validation set?

5 DCF and minDCF evaluation

Analyze the performance of the MVG classifier and its variants for different applications. Start by considering five applications, given by (π_1, C_{fn}, C_{fp}):

1. (0.5, 1.0, 1.0), i.e., uniform prior and costs
2. (0.9, 1.0, 1.0), i.e., the prior probability of a genuine sample is higher (in our application, most users are legit)
3. (0.1, 1.0, 1.0), i.e., the prior probability of a fake sample is higher (in our application, most users are impostors)
4. (0.5, 1.0, 9.0), i.e., the prior is uniform (same probability of a legit and fake sample), but the cost of accepting a fake image is larger (granting

access to an impostor has a higher cost than labeling as impostor a legit user - we aim for strong security)

5. (0.5, 9.0, 1.0), i.e., the prior is uniform (same probability of a legit and fake sample), but the cost of rejecting a legit image is larger (granting access to an impostor has a lower cost than labeling a legit user as impostor - we aim for ease of use for legit users)

Represent the applications in terms of effective prior. What do you obtain? Observe how the costs of mis-classifications are reflected in the prior: stronger security (higher false positive cost) corresponds to an equivalent lower prior probability of a legit user.

We now focus on the three applications, represented in terms of effective priors (i.e., with costs of errors equal to 1) given by $\pi = 0.1$, $\pi = 0.5$, and $\pi = 0.9$, respectively.

For each application, compute the optimal Bayes decisions for the validation set for the MVG models and its variants, with and without PCA (try different values of m). Compute DCF (actual) and minimum DCF for the different models. Compare the models in terms of minimum DCF. Which models perform best? Are relative performance results consistent for the different applications? Now consider also actual DCFs. Are the models well calibrated (i.e., with a calibration loss in the range of few percents of the minimum DCF value) for the given applications? Are there models that are better calibrated than others for the considered applications?

Consider now the PCA setup that gave the best results for the $\pi = 0.1$ configuration (this will be our main application). Compute the Bayes error plots for the MVG, Tied and Naive Bayes Gaussian classifiers. Compare the minimum DCF of the three models for different applications, and, for each model, plot minimum and actual DCF. Consider prior log odds in the range $(-4, +4)$. What do you observe? Are model rankings consistent across applications (minimum DCF)? Are models well-calibrated over the considered range?

6 Binary logistic regression

We analyze the binary logistic regression model on the project data. We start considering the standard, non-weighted version of the model, without any pre-processing.

Train the model using different values for λ. You can build logarithmic-spaced values for λ using `numpy.logspace`. To obtain good coverage, you can use `numpy.logspace(-4, 2, 13)` (check the documentation). Train the model with each value of λ, score the validation samples and compute the corresponding actual DCF and minimum DCF for the primary application $\pi = 0.1$. To compute actual DCF remember to remove the log-odds of the training set empirical prior. Plot the two metrics as a function of λ (suggestion: use a logarithmic scale for the x-axis of the plot - to change the scale of the x-axis you can use `matplotlib.pyplot.xscale('log', base=10)`). What do you observe? Can

you see significant differences for the different values of λ? How does the regularization coefficient affect the two metrics?

Since we have a large number of samples, regularization seems ineffective, and actually degrades actual DCF since the regularized models tend to lose the probabilistic interpretation of the scores. To better understand the role of regularization, we analyze the results that we would obtain if we had fewer training samples. Repeat the previous analysis, but keep only 1 out of 50 model training samples, e.g., using data matrices `DTR[:, ::50]`, `LTR[::50]` (apply the filter only on the model training samples, not on the validation samples, i.e., after splitting the dataset in model training and validation sets). What do you observe? Can you explain the results in this case? Remember that lower values of the regularizer imply a larger risk of overfitting, while higher values of the regularizer reduce overfitting, but may lead to underfitting and to scores that lose their probabilistic interpretation.

In the following, we will again consider only the full dataset. Repeat the analysis with the prior-weighted version of the model (remember that, in this case, to transform the scores to LLRs you need to remove the log-odds of the prior that you chose when training the model). Are there significant differences for this task? Are there advantages using the prior-weighted model for our application (remember that the prior-weighted model requires that we know the target prior when we build the model)?

Repeat the analysis with the quadratic logistic regression model (again, full dataset only). Expand the features, train and evaluate the models (you can focus on the standard, non prior-weighted model only, as the results you would obtain are similar for the two models), again considering different values for λ. What do you observe? In this case is regularization effective? How does it affect the two metrics?

The non-regularized model is invariant to affine transformations of the data. However, once we introduce a regularization term, affine transformations of the data can lead to different results. Analyze the effects of centering (optionally, you can also try different strategies, including Z-normalization and whitening, as well as PCA) on the model results. You can restrict the analysis to the linear model. Remember that you have to center both datasets with respect to the model training dataset mean, i.e., you must not use the validation data to estimate the pre-processing transformation. For this task, you should observe only minor variations, as the original features were already almost standardized.

As you should have observed, the best models in terms of minimum DCF are not necessarily those that provide the best actual DCFs, i.e., they may present significant mis-calibration. We will deal with score calibration at the end of the course. For the moment, we focus on selecting the models that optimize the minimum DCF on our validation set. Compare all models that you have trained up to now, including Gaussian models, in terms of minDCF for the target application $\pi = 0.1$. Which model(s) achieve(s) the best results? What kind of separation rules or distribution assumptions characterize this / these model(s)? How are the results related to the characteristics of the dataset features?

Suggestion for upcoming laboratories: the last laboratories will cover score calibration, and will require to evaluate the results of the models that you tested on a held-out evaluation set. We suggest that you save the models, and the corresponding validation scores as well, since these will be required for score calibration (you can skip the models trained with the reduced dataset, as they won't be needed).

7 SVM Application to the dataset

NOTE: training SVM models may require some time. To speed-up the process, we suggest that you first setup the experiments for this project using only a fraction of the model training data. Once the code is ready, you can then re-run all the experiments with the full dataset.

Apply the SVM to the project data. Start with the linear model (to avoid excessive training time we consider only the models trained with $K = 1.0$). Train the model with different values of C. As for logistic regression, you should employ a logarithmic scale for the values of C. Reasonable values are given by `numpy.logspace(-5, 0, 11)`. Plot the minDCF and actDCF ($\pi_T = 0.1$) as a function of C (again, use a logarithmic scale for the x-axis). What do you observe? Does the regularization coefficient significantly affect the results for one or both metrics (remember that, for SVM, low values of C imply strong regularization, while large values of C imply weak regularization)? Are the scores well calibrated for the target application? What can we conclude on linear SVM? How does it perform compared to other linear models? Repeat the analysis with centered data. Are the result significantly different?

We now consider the polynomial kernel. For simplicity, we consider only the kernel with $d = 2, c = 1$ (but better results may be possible with different configurations), and we set $\xi = 0$, since the kernel already implicitly accounts for the bias term (due to $c = 1$). We also consider only the original, non-centered features (again, different pre-processing strategies may lead to better results). Train the model with different values of C, and compare the results in terms of minDCF and actDCF. What do you observe with quadratic models? In light of the characteristics of the dataset and of the classifier, are the results consistent with previous models (logistic regression and MVG models) in terms of minDCF? What about actDCF?

For RBF kernel we need to optimize both γ and C (since the RBF kernel does not implicitly account for the bias term we set $\xi = 1$). We adopt a grid search approach, i.e., we consider different values of γ and different values of C, and try all possible combinations. For γ we suggest you analyze values $\gamma \in \{e^{-4}, e^{-3}, e^{-2}, e^{-1}\}$, while for C, to avoid excessive time but obtain a good coverage of possible good values we suggest log-spaced values `numpy.logspace(-3, 2, 11)` (of course, you are free to experiment with other values if you so wish). Train all models obtained by combining the values of γ and of C. Plot minDCF and actDCF as a function of C, with a different line for each value of γ (i.e., four lines for minDCF and four lines for actDCF). Analyze

the results. Are there values of γ and C that provide better results? Are the scores well calibrated? How do the results compare to previous models? Are there characteristics of the dataset that can be better captured by RBF kernels?

Optional: Consider again the polynomial kernel, but with $d = 4, c = 1, \xi = 0$. Train the model with different values of C (use again `numpy.logspace(-5, 0, 11)`), and compare the results in terms of minDCF and actDCF. What do you observe with quadratic models? Can you explain the better results in terms of the characteristics of the dataset? To answer, consider only the last two features of each sample (look at the scatter plots) $y_i = x_{i,[4:6]}$. Consider how these features would be transformed by a simple degree 2 kernel that maps each sample to $y_i \rightarrow z_i = y_{i,[0:1]} y_{i,[1:2]}$ (suggestion: draw a few samples on paper, one per cluster). Then consider which kind of separation surfaces would be required to separate the z_i 1-D feature vectors — remember that in 1-D linear rules correspond to a sample being on either side of a threshold, quadratic rules correspond to inequalities that involve the sample being inside a (possibly empty) interval, and so on.